Many-core on the Grid: From exploration to production

2014 J. Phys.: Conf. Ser. 513 052037

(http://iopscience.iop.org/1742-6596/513/5/052037)

Journal of Physics: Conference Series **513** (2014) 052037 doi:10.1088/1742-6596/513/5/052037

Many-core on the Grid: From exploration to production

T Doherty[1], M Doidge[2], T Ferrari[3], L Salvador[1], J Walsh[4], A Washbrook[5]

[1] Kelvin Building, Physics and Astronomy, University of Glasgow, Glasgow, G12 8QQ, United Kingdom
[2] Physics Department, Lancaster University, Lancaster. LA1 4YB, United Kingdom
[3] EGI.eu, Science Park 140, 1098 XG Amsterdam, The Netherlands
[4] School of Computer Science and Statistics, Trinity College Dublin, Ireland
[5] SUPA, School of Physics and Astronomy, The University of Edinburgh, James Clerk Maxwell Building, Mayfield Road, Edinburgh, EH9 3JZ, United Kingdom

E-mail: awashbro@ph.ed.ac.uk

Abstract. High Energy Physics experiments have successfully demonstrated that many-core devices such as GPUs can be used to accelerate critical algorithms in their software. There is now increasing community interest for many-core devices to be made available on the LHC Computing Grid infrastructure. Despite anticipated usage there is no standard method available to run many-core applications in distributed computing environments and before many-core resources are made available on the Grid a number of operational issues such as job scheduling and resource discovery will need to be addressed. The key challenges for Grid-enabling many-core devices will be discussed.

1. Introduction

Feasibility studies performed by the HEP community have shown that many-core devices such as GPUs have generated significant timing performance improvements for trigger systems and data analysis software [1],[2]. To provide many-core support to interested communities using Grid resources it is envisaged that middleware technologies such as LRMS integration, resource matching and allocation, CPU accounting and information publishing will need to be modified to manage access to these new types of computational resources. For example, software developed for many-core devices is often optimised for a particular hardware model and therefore specific architecture details - such as the shared memory capacity and the maximum allocatable thread block size - will be mandatory for efficient job resource matching.

In this study we investigate the steps that will need to be taken in order for many-core devices to be included in a Grid production environment. GPU devices available at Tier-2 Grid sites in the UK will be used to demonstrate the configuration steps required to make many-core resources available.

2. Site configuration and experience

The Edinburgh (ECDF) and Lancaster UK Tier-2 Grid sites were selected to demonstrate many-core access on the Grid. Both computing sites have Nvidia Tesla GPUs available as part of their

infrastructure and the properties of these hosted device is shown in Table 1. To accept many-core jobs each site made the following common configuration changes:

- A new batch system queue was defined to partition many-core jobs from normal CPU-based workload. Jobs were given exclusive access to worker node resources to avoid concurrent many-core device access by other jobs residing on the same worker node.
- GPU-specific job command flags were prepended to each incoming job script in order for relevant jobs to be routed to the correct queue, to request an appropriate amount of GPU devices, and to prepare the worker node for the execution of CUDA software.
- Nvidia CUDA associated software and drivers was made available on each worker node hosting GPU devices.
- Many-core queues defined at candidate sites were validated by test jobs to ensure that these were available to any authorised and authenticated Grid user.

A common issue encountered by both sites was determining a method to overcome differences in many-core hardware and their associated runtime environments. In particular for GPUs the choice of CUDA version to install was difficult to ascertain. Support for only the latest version of CUDA would exclude the ability to run workload based on earlier versions of the CUDA API whilst older versions may limit the extended functionality available on newer generation devices.

GPU Model (Tesla)	**T10**	**M2050**	**M2075**	**C1060**	**K20c**
CUDA Capability	2.0	1.3	2.0	1.3	3.5
Multiprocessors	14	30	14	8	192
Number of cores	448	240	448	240	2496
Global memory (MB)	2687	4096	5375	4096	4800
Constant memory (kB)	64	64	64	64	64
Shared memory (kB)	48	16	48	16	48
Registers per block	32768	16384	32768	16384	65536
Max threads per MP	1536	1024	1536	1024	2048
Max threads per block	1024	512	1024	512	1024

Table 1. Many-core device specifications for GPUs available at the candidate UK Tier-2 sites. Two additional GPUs (C1060 and K20c) which were used for comparison testing are also listed.

3. Functionality and performance testing

A series of benchmark tests were defined to check the functionality and performance of the many-core devices hosted at the candidate Tier-2 Grid sites. Tests were derived from a subset of code examples available in the Nvidia CUDA Software Development Kit [3] and a steering script was created to re-compile the code examples on the worker nodes. An additional option was created to capture profiling information gathered by CUDA profiling tools [4]. The results from the host and device bandwidth rate tests and matrix multiplication performance tests are shown in Figure 1. Note that the variation across devices shown in these figures indicates that performance will vary for identical workloads even for GPUs of similar capability.

3.1. EPIC VO many-core testing

The EPIC project [5] was chosen as an example Virtual Organisation (VO) to demonstrate realistic use of many-core based workload on the Grid. The project is developing epidemiological models using the life histories of cattle and the location of farms to understand the dynamics

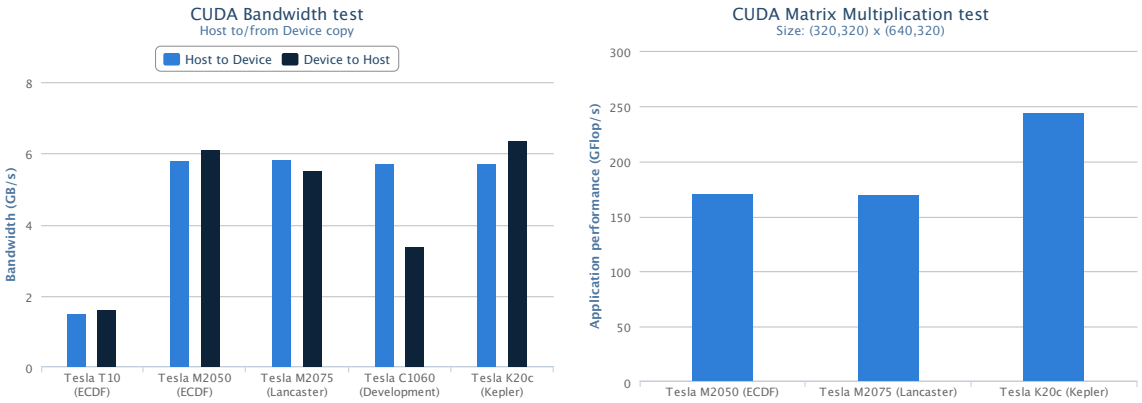

Figure 1. Device bandwidth and matrix multiplication test results for GPU devices available at candidate Tier-2 sites.

of bovine tuberculosis transmission. The spatial analysis techniques used in their analysis to model thousands of individual agents is well suited to parallelisation using many-core devices.

Job submission from the EPIC VO was enabled at both Grid sites and their CUDA-based software was locally installed for execution on the worker node. It was immediately observed that their code had dependencies on an older version of CUDA which had a clear impact on portability with the selection of GPU devices used in the study. Work is ongoing by project developers and site administrators to enable their code to be optimised on the many-core devices available.

4. Many-core Advertising and Accounting

Now that many-core functionality testing and real world use cases are starting to be validated on the Grid it is possible to determine how many-core devices can be advertised for more general use.

4.1. Information Publishing

It is not expected that many-core jobs submitted to the grid will be able to run on every type of many-core device. As discussed in Section 3, this may be due to compilation dependencies or requirements on device functionality. Furthermore, user code may be heavily optimised for a particular many-core device and perform poorly when executed on other devices. It will therefore be useful to extend the information publishing schema to cover the entire capability range so that jobs can be allocated appropriate resources. For example, the properties gathered from the GPU functionality testing is shown in Table 1 could be used to match resources given an adequate description of job requirements. The Lancaster Tier-2 Grid site created a local GLUE [6] schema variable to store the software runtime environment information as part of their site configuration. Job requirements were identified within the local submission scripts so that they could be submitted to the many-core queue and request the correct resources for execution.

4.2. Job Accounting

Job accounting middleware could also be extended to store accurate information on the utilisation of many-core devices. For example, many-core processing time could be used to determine the "efficiency" of a job either in relation to the total CPU time or to the overall

wallclock time. As a demonstration the profiling tools used in Section 3 were used to measure a number of observables of interest including GPU processing time and the occupancy of the GPU device for each kernel method call made during code execution. The timing and occupancy values for a sample profiled job on GPU devices available in this study is shown in Table 2. Kernel efficiency is derived from the ratio of CPU time to GPU time and occupancy is an average of the device occupancy reported for each kernel method call. This latter property could be used as secondary measure of job efficiency to determine time-independent device utilisation. Although these are subjective measures they can highlight how much time the GPU is active when computation is being offloaded to the device which can be useful feedback to developers and computing sites who wish to maximise resource delivery.

GPU model (Tesla)	**GPU time**	**CPU time**	**Kernel Efficiency**	**Occupancy**
T10	85.2	0.44	99.47%	99.95%
M2075	24.4	0.050	99.80%	80.50%
C1060	32.8	0.039	99.88%	99.97%
K20c	23.5	0.065	99.72%	100%

Table 2. Sample profiling results from a CUDA-based asynchronous streaming test (CPU and GPU time in ms). GPU time is defined as the execution time for the GPU kernel or memory copy method and CPU time is defined as the overhead incurred to launch the kernel. The timing values listed have been summed over each kernel invocation.

5. Outlook

The many-core configuration and testing performed in this study has highlighted deployment issues that are not present when providing CPU resources to the Grid, such as the dependence on device functionality. Effort will therefore be required to provide a suitable method to allocate many-core resources based on an extended information publishing schema that will cover all compatibility constraints. It has been shown that information to populate such a schema is readily available through functional and performance tests, and that additional information from job profiling could be used to generate additional accounting metrics.

The EGI GPGPU working group will develop a GLUE 2.0 execution environment definition for GPGPU resources. This is intended to succinctly describe the feature and state of GPGPU resources:

- Defining baseline device properties from vendor independent tools
- Runtime environment conventions
- Batch system integration
- Dynamic information such as availability of GPGPU resources

Any concerted effort to provide a general solution to include many-core devices into Grid operations would also require support for Many Integrated Core devices (MICs) such as the Intel Xeon Phi. Although code portability is expected to involve less overhead than GPU-based code there are other factors that have to be considered. The Xeon Phi runs an operating system residing on the device RAM which is distinct from the host and will not automatically have access to the full worker node environment. Access to VO software, external libraries and package dependencies will all need to be assessed and potentially included as part of the device configuration. A MIC has now been deployed at the ECDF Tier-2 site and is being evaluated for integration as a Grid resource for future many-core workloads from High Energy Physics experiments.

References

[1] P J Clark et al 2011 *J. Phys.: Conf. Series* **331** 022031
[2] A Hoecker et al 2012 *J. Phys.: Conf. Series* **396** 022055
[3] CUDA GPU computing SDK https://developer.nvidia.com/gpu-computing-sdk
[4] CUDA profiler users guide http://docs.nvidia.com/cuda/profiler-users-guide/index.html
[5] The Scottish Government's Centre of Expertise on Animal Disease Outbreaks http://www.sruc.ac.uk/epic/
[6] S Burke et al 2010 *J. Phys.: Conf. Series* **219** 062005